Mac OS X Color Image Management Explained

Session 209

These are confidential sessions—please refrain from filming or taking pictures.

Graphics and Media

Mac OS X Color Image Management Explained

David Hayward
Software Engineer

http://developer.apple.com/wwdc2005/

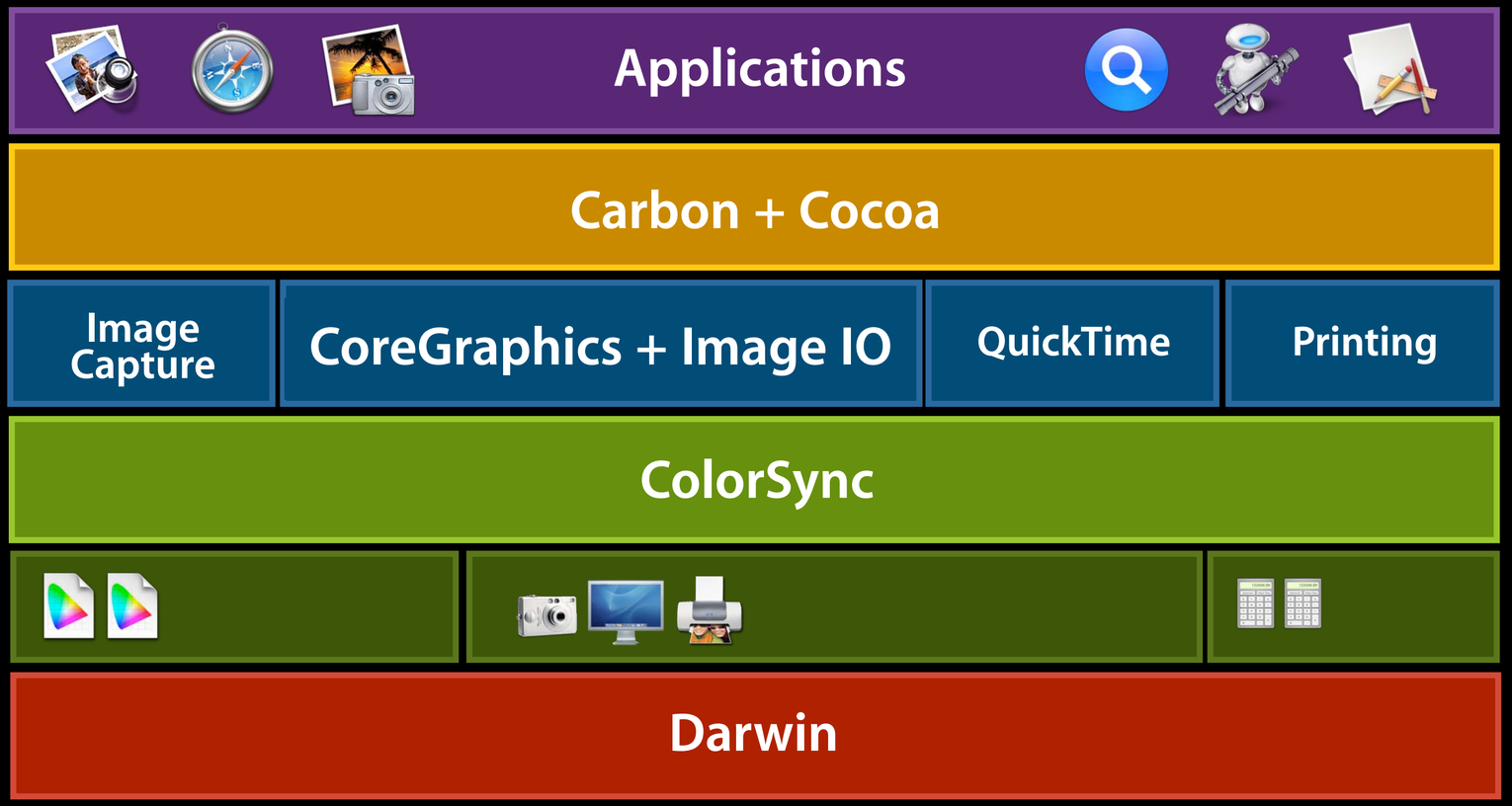
Introduction

- Architectural Overview
- Opening Images with Image IO
- Displaying Images
- Filtering Images with Core Image
- Printing Images
- Saving Images with Image IO

Architectural Overview

http://developer.apple.com/wwdc2005/

Architectural Overview



API Data Types for Color Management

	Color Space	Bitmap Image	Spot Color
Cocoa	NSColorSpace	NSBitmapImageRep	NSColor
Quartz	CGColorSpace	CGImage	CGColor
ColorSync	CMProfile	CMBitmap	CMColor

API Data Types for Color Management

	Color Space	Bitmap Image	Spot Color
Cocoa	NSColorSpace	NSBitmapImageRep	NSColor
Quartz	CGColorSpace	CGImage	CGColor
ColorSync	CMProfile	CMBitmap	CMColor

API Data Types for Color Management

	Color Space	Bitmap Image	Spot Color
Cocoa	NSColorSpace	NSBitmapImageRep	NSColor
Quartz	CGColorSpace	CGImage	CGColor
ColorSync	CMProfile	CMBitmap	CMColor

API Data Types for Color Management

	Color Space	Bitmap Image	Spot Color
Cocoa	NSColorSpace	NSBitmapImageRep	NSColor
Quartz	CGColorSpace	CGImage	CGColor
ColorSync	CMProfile	CMBitmap	CMColor

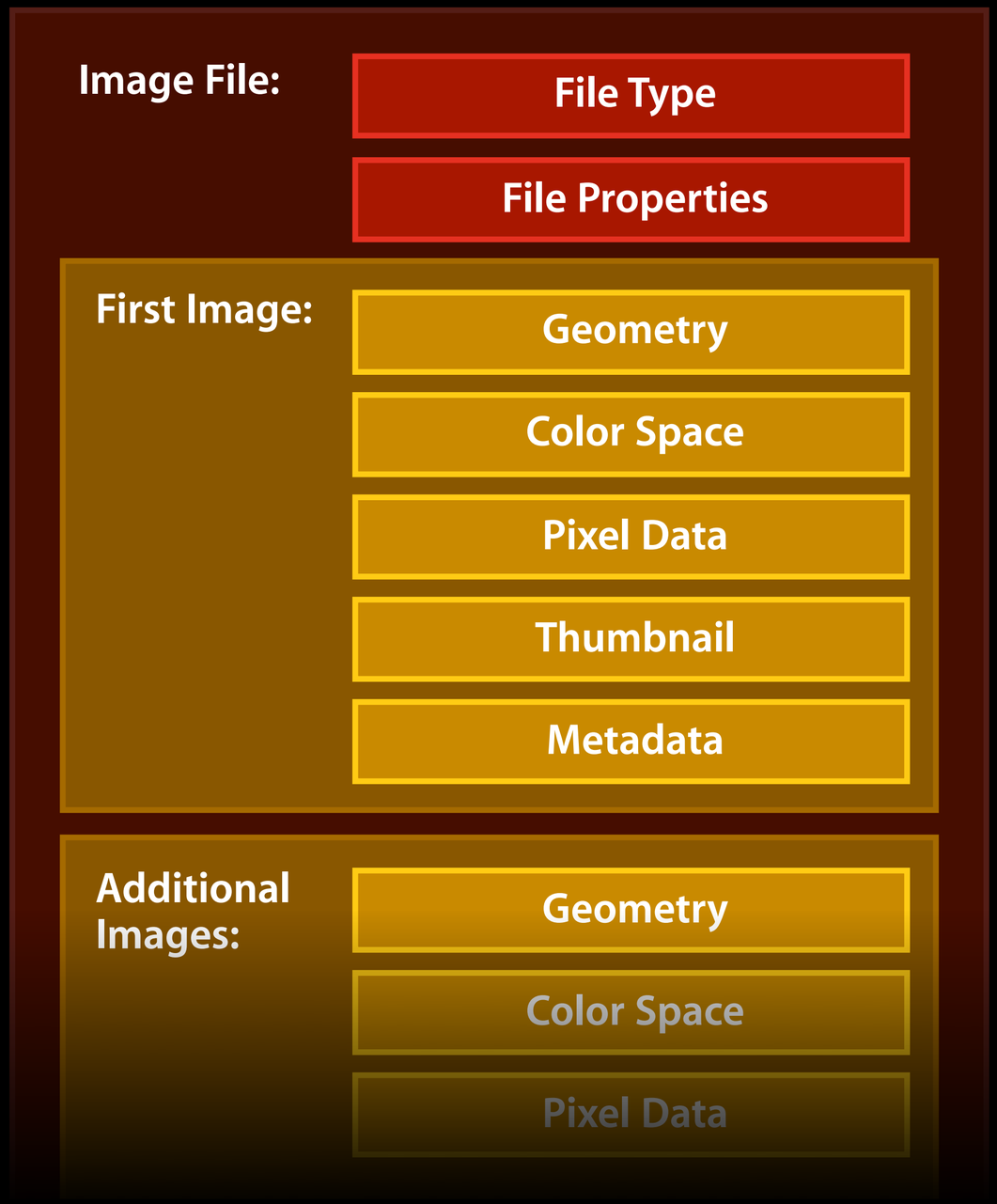
Image IO Features

- Read many file formats
- Write to file formats
- Read and write metadata
- Incremental loading
- Floating point support
- Automatic Color Management

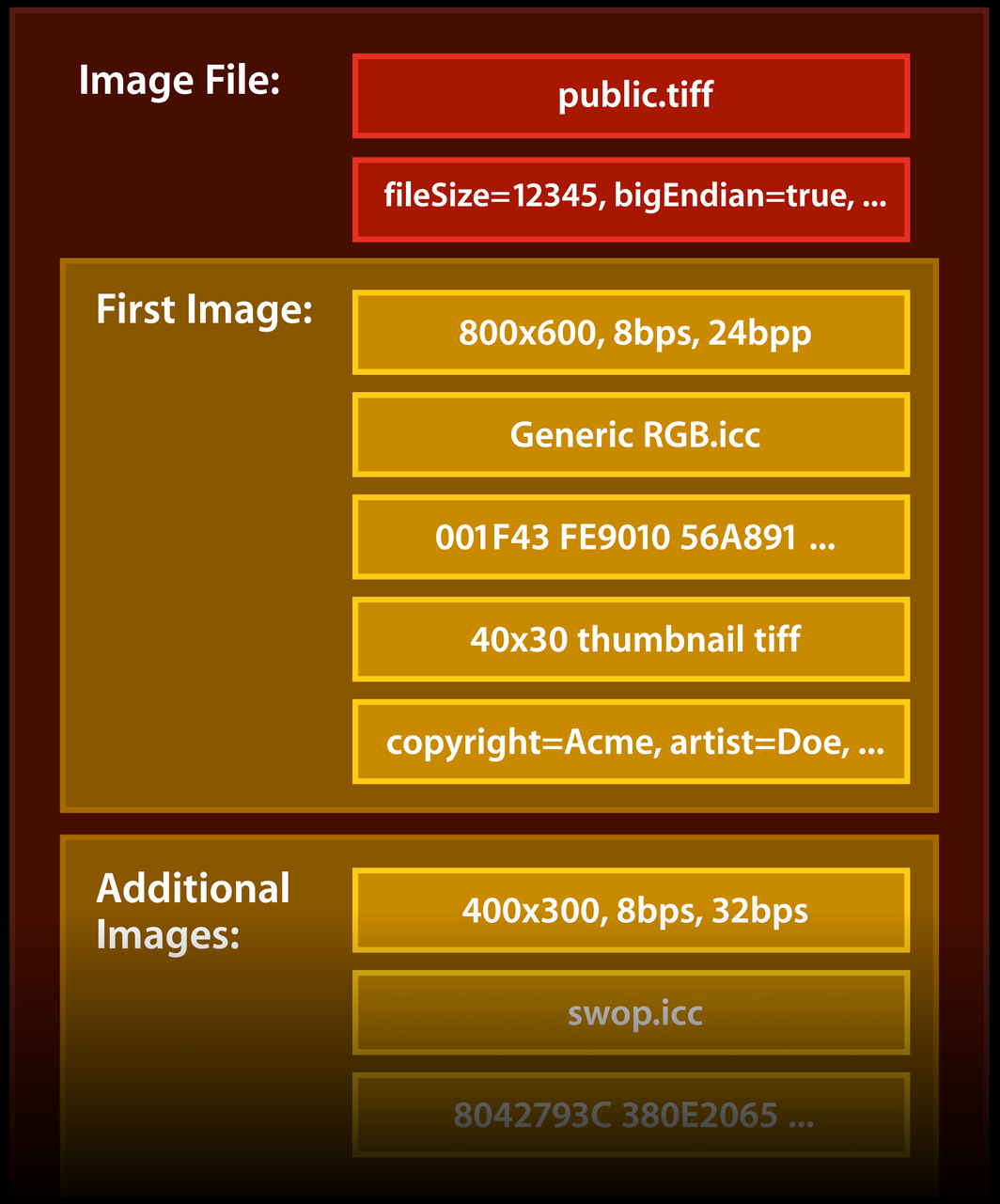
Image IO Supported Formats

- Web standards
  - TIFF, JPEG, PNG, GIF, JPEG-2000
- Floating Point HDR formats
  - OpenEXR, Radiance, LogLuv TIFF, Float TIFF, and Pixar TIFF
- Camera Raw formats
  - Canon, Nikon, Minolta, Olympus, Sony
- Others
  - .bmp, .psd, .qtif, .tga, .sgi, .xbm, .icns, .ico, .pict, .ptng, .fpx
- Metadata
  - Exif, IPTC, CIFF, GPS, some MakerNotes, …
- More to come…

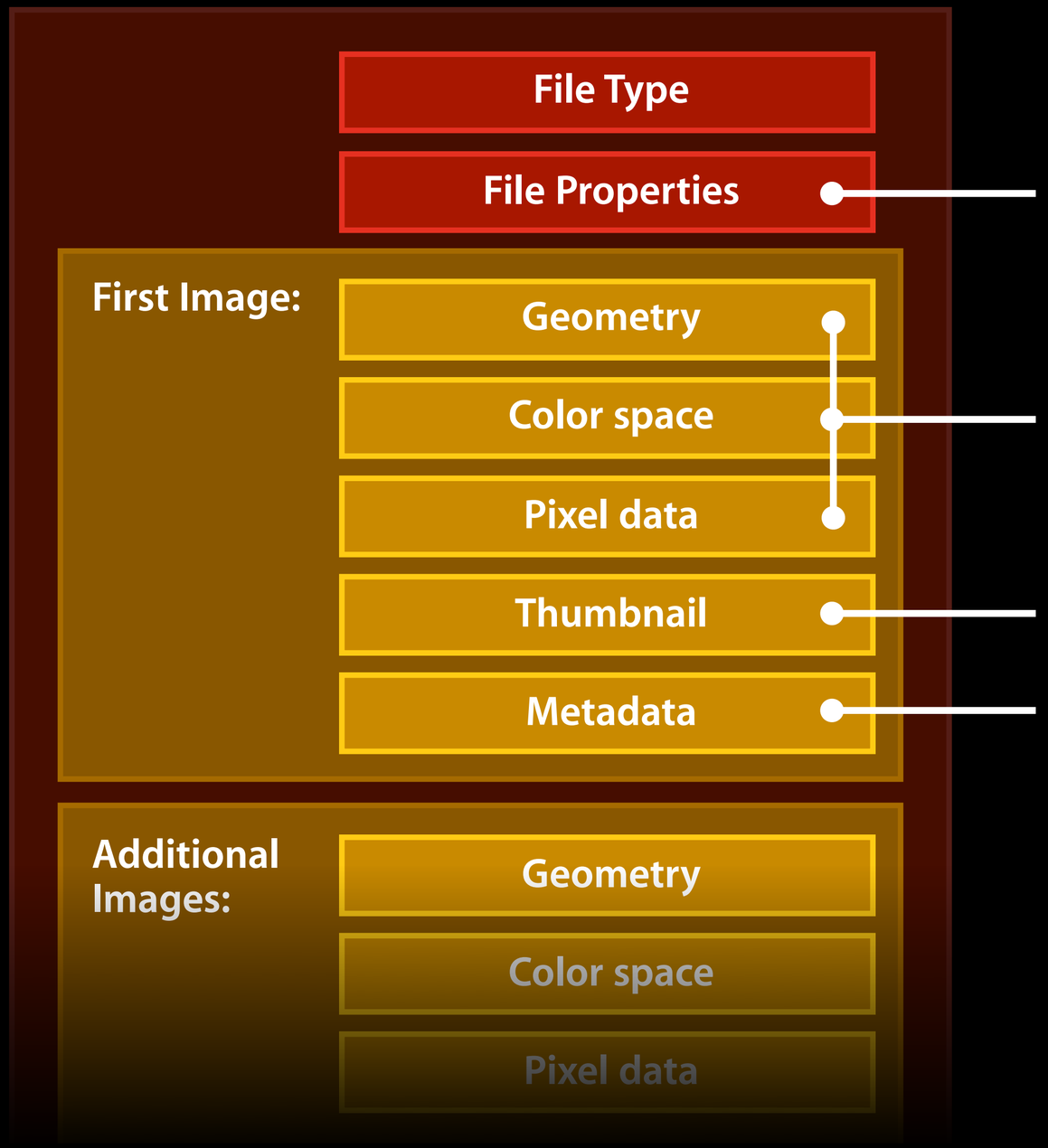
What's in an Image File



What's in an Image File



What's in an Image File

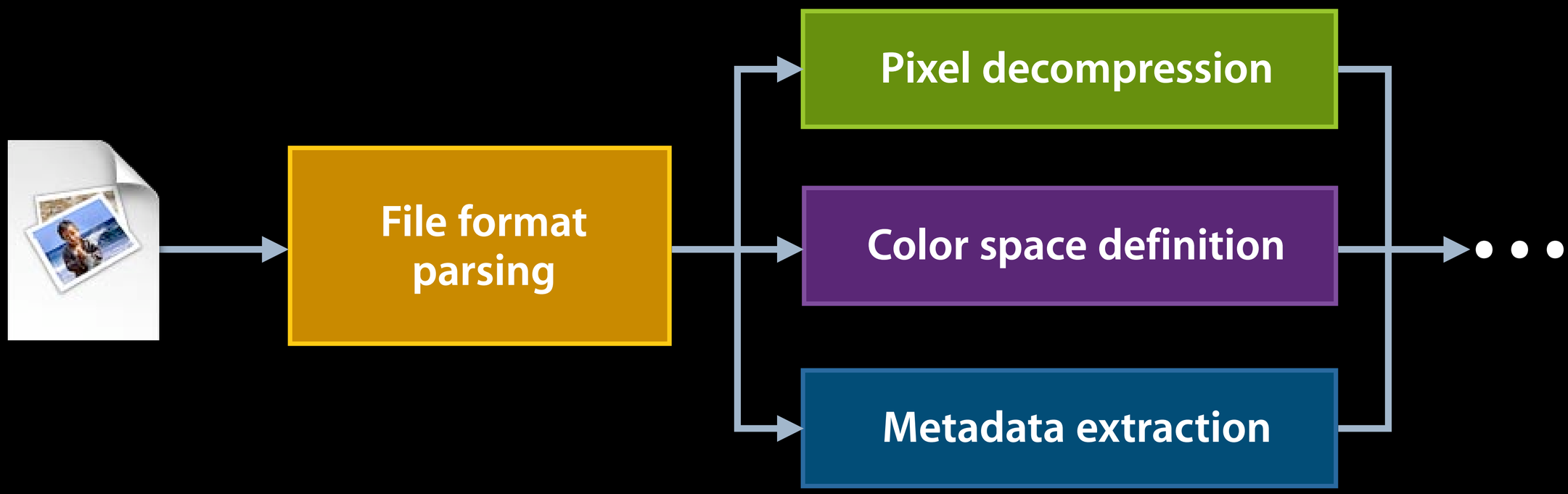


Opening Images With Image IO

http://developer.apple.com/wwdc2005/

Reading Digital Images

General steps when loading an image



Opening Images With Image IO

Basics

- Create a CGImageSource
  - Use `CGImageSourceCreateWithURL`
    or `CGImageSourceCreateWithData`
    or `CGImageSourceCreateWithDataProvider`
- Get Image Properties
  - Type identifier: `CGImageSourceGetType`
  - Image count: `CGImageSourceGetCount`
- For each
  - Get image: `CGImageSourceCreateImageAtIndex`
  - Get metadata: `CGImageSourceCopyPropertiesAtIndex`
  - Get thumb: `CGImageSourceCreateThumbnailAtIndex`

Opening Images With Image IO

CGImageSource image and metadata

```
CGImageRef GetFirstImage (CFURLRef url, float* xdpi, float* ydpi, int* orientation)
{
   CGImageSourceRef isrc = CGImageSourceCreateWithURL(url, nil);

   CFDictionaryRef props = CGImageSourceCopyPropertiesAtIndex(isrc,0,nil);
   CFNumberRef val;

   val = CFDictionaryGetValue(props, kCGImagePropertyDPIWidth);
   if (val) CFNumberGetValue(val, kCFNumberFloatType, xdpi);

   val = CFDictionaryGetValue(props, kCGImagePropertyDPIHeight);
   if (val) CFNumberGetValue(val, kCFNumberFloatType, ydpi);

   val = CFDictionaryGetValue(props, kCGImagePropertyOrientation);
   if (val) CFNumberGetValue(val, kCFNumberIntType, orientation);

   return CGImageSourceCreateImageAtIndex(isrc, 0, nil);
}
```

Opening Images With Image IO

CGImageSource thumbnail

```
CGImageRef GetFirstThumb (CFURLRef url)
{
   CGImageSourceRef isrc = CGImageSourceCreateWithURL(url, nil);

   CFStringRef keys[2]; CFTypeRef vals[2];

   keys[0] = kCGImageSourceCreateThumbnailFromImageIfAbsent;
   vals[0] = kCFBooleanTrue;

   keys[1] = kCGImageSourceThumbnailMaxPixelSize;
   vals[1] = CreateIntNumber(160);

   CFDictionaryRef opts = CFDictionaryCreate(nil, keys, values, 2,
       &kCFCopyStringDictionaryKeyCallBacks, &kCFTypeDictionaryValueCallBacks);

   return CGImageSourceCreateThumbnailAtIndex(isrc, 0, opts);
}
```

Opening Images With Image IO

Advanced Techniques

- Loading Images Incrementally
- Reading Floating Point HDR Images
- Dynamic support for all Image IO readable types

Opening Images: Advanced Techniques

Loading an image incrementally

```
void DrawFirstImageIncrementally (CFURLRef url)
{
   CGImageSourceRef isrc = CGImageSourceCreateIncremental(nil);
   do {
       bool       done;
       CFDataRef  data = GetAccumulatedDataSoFar(&done)

       CGImageSourceUpdateData(isrc, data, done);

       CGImageRef img = CGImageSourceCreateImageAtIndex(isrc, 0, nil);
       CGImageSourceStatus status = CGImageSourceGetStatusAtIndex(isrc, 0);
       if (img) {
           DrawImage(img);
           CFRelease(img);
       }
   } while (done==false && status!=kCGImageStatusComplete);
}
```

Opening Images: Advanced Techniques

Reading floating point images

- Some file formats support only one pixel depth
  - e.g. a JPEG file is always 8 bits per sample
- Some file formats support several pixel depths
  - e.g. a TIFF file can be 1,2,4,8, or 16 bits per sample

- As a rule, a CGImageRef returned by Image IO will be the same depth as the file

Opening Images: Advanced Techniques

Reading floating point images

- HDR files are a bit more complicated
  - The data in the file is typically specially packed
  - There are often several methods to unpack the bits
    - Unpacked to floats (32 or 16) or integers (16 or 8)
    - Unpacked to extended range (<0 .. >1) or compressed (0..1)

- By default, the CGImageRef returned by Image IO will be compressed to a 16bit integers
- By request, the CGImageRef can be returned as extended range floats

Opening Images: Advanced Techniques

Reading floating point images

```
CGImageRef GetFirstImageFloat (CFURLRef url, Boolean* isFloat)
{
   CGImageSourceRef isrc = CGImageSourceCreateWithURL(url, nil);

   CFStringRef keys[1] = { kCGImageSourceShouldAllowFloat };
   CFTypeRef values[1] = { kCFBooleanTrue };

   CFDictionaryRef opts = CFDictionaryCreate(nil, keys, values, 1,
       &kCFCopyStringDictionaryKeyCallBacks, &kCFTypeDictionaryValueCallBacks);

   CFDictionaryRef props = CGImageSourceCopyPropertiesAtIndex(isrc,0,opts);

   *isFloat = CFDictionaryContainsKey(props, kCGImagePropertyIsFloat);

   return CGImageSourceCreateImageAtIndex(isrc, 0, opts);
}
```

Opening Images: Advanced Techniques

Dynamic support for all Image IO readable types

- Get array of Image IO readable types: `CGImageSourceCopyTypeIdentifiers`
- Use `UTTypeCopyDeclaration` to get extensions of each readable type

ImageApp Sample Code

http://developer.apple.com/wwdc2005/

Displaying Images

Luke Wallis
Software Engineer

http://developer.apple.com/wwdc2005/

Displaying Digital Images

General steps when displaying an image

Drawing Images in Quartz

CGContextDrawImage call

```
void CGContextDrawImage(
   CGContextRef context, // The graphics context in which to draw the image.
   CGRect       rect,    // The location and dimensions in user space of the
                         // bounding box in which to draw the image.
   CGImageRef   image    // The image to draw.
);
```

Quartz Graphics Context

The abstract for a drawing destination

- Contains drawing parameters and all device-specific information
- Can represent a window, a bitmap, an OpenGL surface, a PDF document, or a PostScript document

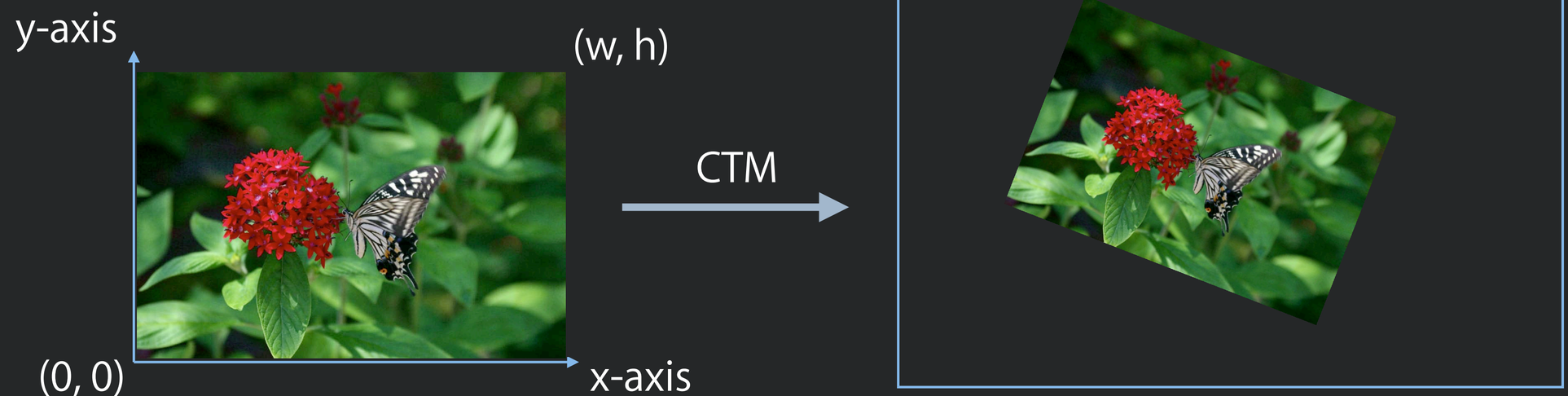
See “Quartz 2D Programming Guide” for more details

Drawing Images in Quartz

Quartz 2D Coordinates

User Space
w = image width
h = image height

Device Space
w′ = window context width
h′ = window context height

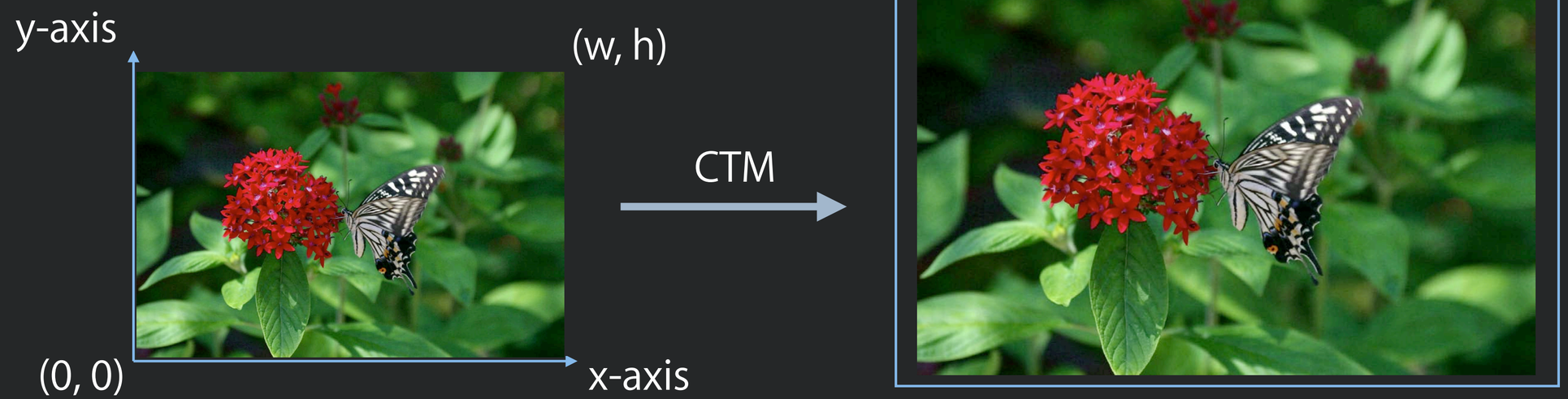


Drawing Images in Quartz

Quartz 2D Coordinates

User Space
w = image width
h = image height

Device Space
w' = window context width
h' = window context height



Drawing in a Cocoa NSView

Using the CGContextRef of a NSView

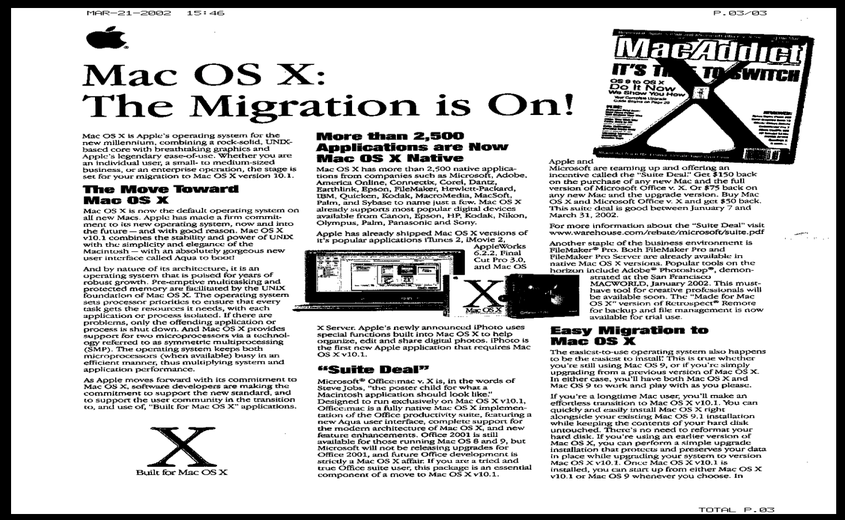
```
- (void) drawRect:(NSRect)rect
{
   CGContextRef context = [[NSGraphicsContext currentContext] graphicsPort];

   size_t w = CGImageGetWidth(mImage)
   size_t h = CGImageGetHeight(mImage);

   CGRect rect = {{0,0},{w,h}};

   CGAffineTransform ctm = [self imageTransformForView];

   CGContextConcatCTM(context, ctm);

   CGContextDrawImage(context, imageRect, mImage);
}
```

Drawing Images With Quartz

Image orientation

Drawing Images With Quartz

Aspect ratio



Drawing Images With Quartz

Respecting image orientation and aspect ratio

```
- (CGAffineTransform) imageTransform {
   float x = ([self dpiH]>[self dpiW]) ? [self dpiH]/[self dpiW] : 1;
   float y = ([self dpiW]>[self dpiH]) ? [self dpiW]/[self dpiH] : 1;
   float w = x * CGImageGetWidth(mImage), h = y * CGImageGetHeight(mImage);

   CGAffineTransform ctms[8] = {
      { x, 0, 0, y, 0, 0},  // 1 = normal
      {-x, 0, 0, y, w, 0},  // 2 = flip horizontal
      {-x, 0, 0,-y, w, h},  // 3 = rotate 180
      { x, 0, 0,-y, 0, h},  // 4 = flip vertical
      { 0,-x,-y, 0, h, w},  // 5 = rot -90, flip vert
      { 0,-x, y, 0, 0, w},  // 6 = rot 90
      { 0, x, y, 0, 0, 0},  // 7 = rot 90, flip vert
      { 0, x,-y, 0, h, 0}}; // 8 = rotate -90

   return ctms[[self orientation]-1];
}
```

Drawing Images: Advanced Techniques

- Extracting ARGB data from an image
- Assigning Default Profiles
- Finding a ColorSync profile for a NSWindow
- Matching across multiple displays
- NSWindowScreenChanged notification

Drawing Images: Advanced Techniques

Extracting ARGB from an image

```
void* GetImageAsARGB (CGImageRef image)
{
   size_t w = CGImageGetWidth(image)
   size_t h = CGImageGetHeight(image);
   CGRect rect = {{0,0},{w,h}};
   void* data = calloc(w*4,h);

   CGColorSpaceRef cs = CGColorSpaceCreateWithName(kCGColorSpaceGenericRGB);

   CGContextRef cgctx = CGBitmapContextCreate(data, w, h,
          8, w*4, cs, kCGImageAlphaPremultipliedFirst);

   CGContextDrawImage(cgctx, rect, image);

   CGColorSpaceRelease(cs);
   CGContextRelease(cgctx);
   return data;
}
```

Drawing Images: Advanced Techniques

Assigning default profiles

```
CGImageRef CopyImageWithDefaultProfile (CGImageRef image) {
   CGColorSpaceRef default = NULL, space = CGImageGetColorSpace (image);
   CGColorSpaceRef gray = CGColorSpaceCreateDeviceGray();
   CGColorSpaceRef rgb = CGColorSpaceCreateDeviceRGB();
   CGColorSpaceRef cmyk = CGColorSpaceCreateDeviceCMYK();
   CGImageRef copy = NULL;

   if (CFEqual (space, gray))      default = mDefaultGray;
   else if (CFEqual (space, rgb))  default = mDefaultRGB;
   else if (CFEqual (space, cmyk)) default = mDefaultCMYK;

   if (default)
       copy = CGImageCreateCopyWithColorSpace(image, default);
   else
       copy = CGImageRetain(image);

   CGColorSpaceRelease(gray); CGColorSpaceRelease(rgb); CGColorSpaceRelease(cmyk);
   return copy;
}
```

Drawing Images: Advanced Techniques

Finding a ColorSync profile for a NSWindow

```
- (CMDisplayIDType) displayID
{
   NSDictionary* d = [[[self window] screen] deviceDescription];
   return (CMDisplayIDType)[[d objectForKey:@"NSScreenNumber"] longValue];
}

- (CMProfileRef) displayProfile
{
   CMProfileRef prof;
   CMGetProfileByAVID([self displayID], &prof);
   return prof;
}

- (CGColorSpaceRef) displayColorSpace
{
   CMProfileRef prof = [self displayProfile];
   CGColorSpaceRef space = CGColorSpaceCreateWithPlatformColorSpace(prof);
   CMCloseProfile(prof);
   return space;
}
```

Drawing Images: Advanced Techniques

Matching across multiple displays

```
- (id) initWithFrame:(NSRect)frameRect
{
   [super initWithFrame:frameRect];

   [[self window] setDisplaysWhenScreenProfileChanges:YES];

   return self;
}
```

Drawing Images: Advanced Techniques

NSWindowDidChangeScreenProfileNotification

```
- (id) initWithFrame:(NSRect)frameRect
{
   [super initWithFrame:frameRect];

   [[self window] setDisplaysWhenScreenProfileChanges:YES];

   [[NSNotificationCenter defaultCenter]
      addObserver: self selector: @selector(newScreenProfile:)
             name: NSWindowDidChangeScreenProfileNotification object:nil];

   return self;
}

- (void) newScreenProfile:(NSNotification*)n
{
   // Do what you wanted to do when the display profile changes
}
```

ImageApp Demo

http://developer.apple.com/wwdc2005/

Filtering Images With Core Image

http://developer.apple.com/wwdc2005/

Drawing Images With Core Image

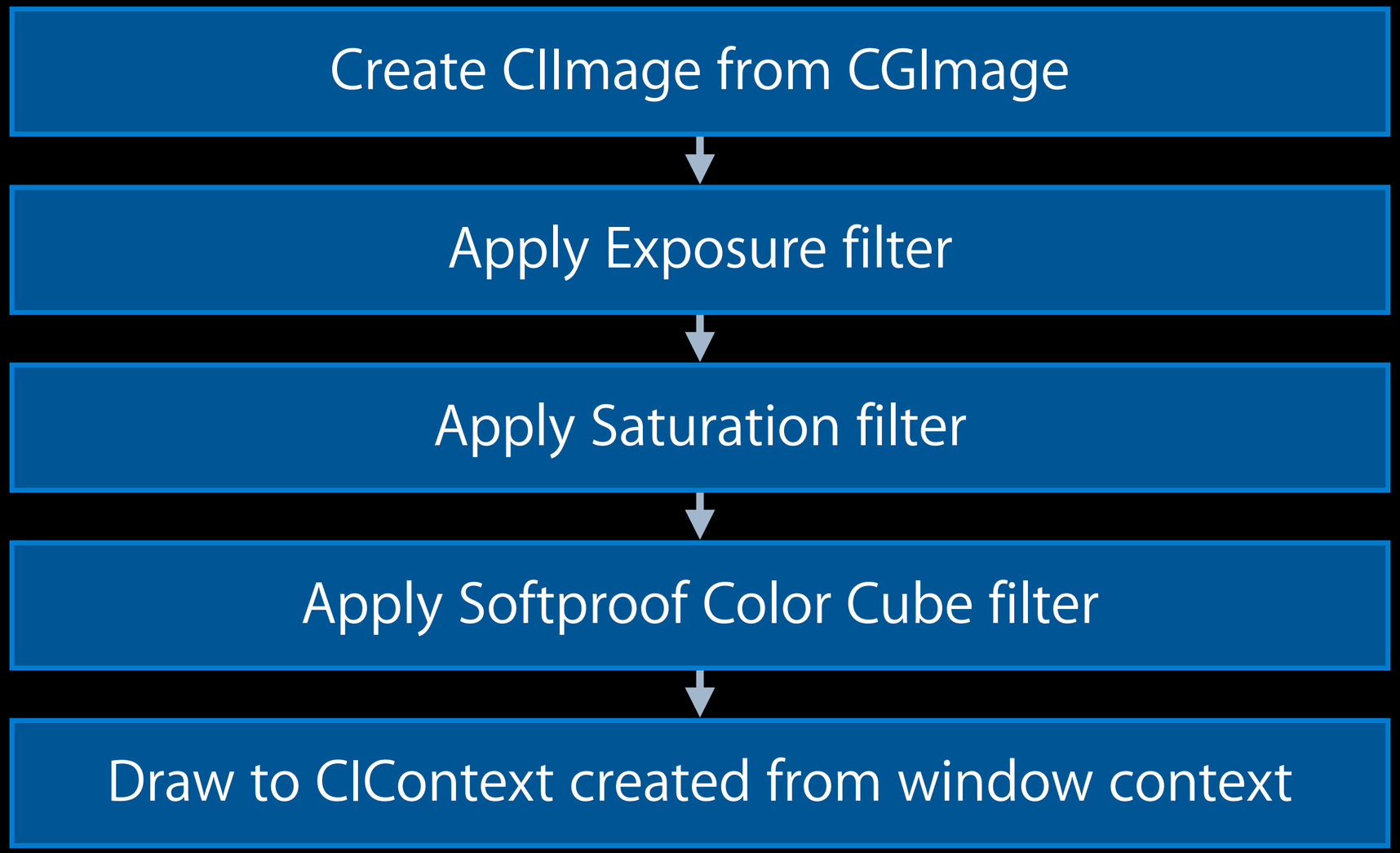
Basics

- Creating a CIImage from a CGImage
- Setting up and applying one or more CIFilters
- Creating a CIContext
- Drawing the result

See “Core Image Programming Guide” for more details

Filtering Images With Core Image

Drawing a filtered image to a window



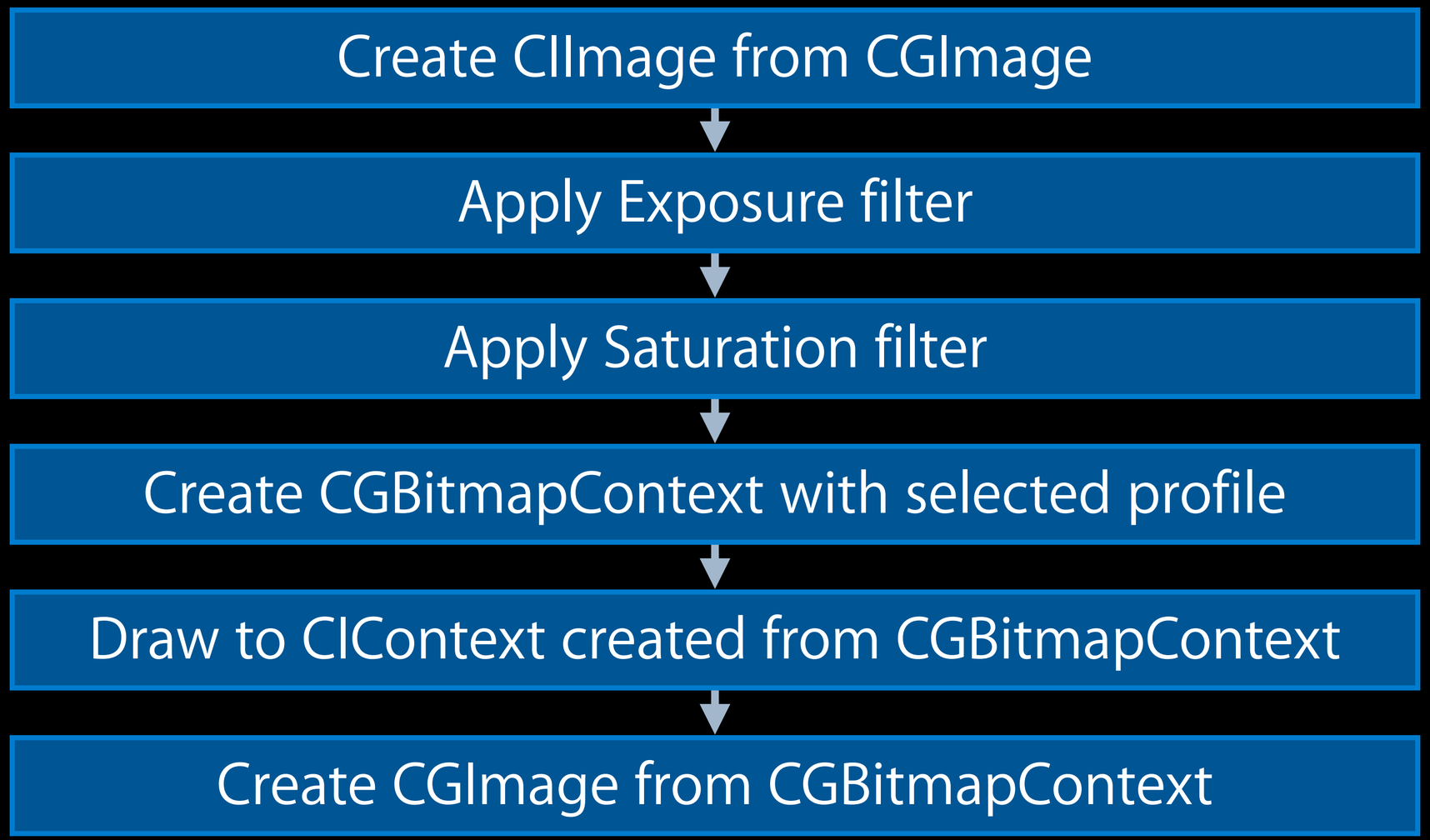
Filtering Images With Core Image

Drawing to a window and softproofing destination profile

```
- (void) drawCIImage
{
   CIImage* ciimg = [CIImage imageWithCGImage:mImage];

   ciimg = [self imageTransformToFitView: ciimg];
   ciimg = [self applyExposureAndSaturationFilters: ciimg];
   ciimg = [self applyColorCubeFilter: ciimg profile:mDestinationProfile];

   CGRect rect = [self centerInViewRect: [ciimg extent]];

   CGContextRef ctx = [[NSGraphicsContextcurrentContext] graphicsPort];

   CIContext* ciContext = [CIContext contextWithCGContext: ctx options:nil];

   [ciContext drawImage: ciimg inRect: rect fromRect: [ciimg extent]];
}
```

Filtering Images With Core Image

Creating a CGImage for printing or saving



Filtering Images With Core Image

Creating a CGImage for printing or saving as an image file

```
- (CGImageRef) createCGImage
{
   CIImage* ciimg = [CIImage imageWithCGImage:mImage];

   ciimg = [self applyExposureAndSaturationFilters: ciimg];

   CGContextRef bmCtx = [self bitmapContext:mDestinationProfile];

   CIContext* ciCtx = [CIContext contextWithCGContext: bmCtx options: nil];

   [ciCtx drawImage: ciimg inRect:mImageRect fromRect: [ciimg extent]];

   CGImageRef image = CGBitmapContextCreateImage(bmCtx);

   CGContextRelease(bmCtx);

   return image;
}
```

Printing Images

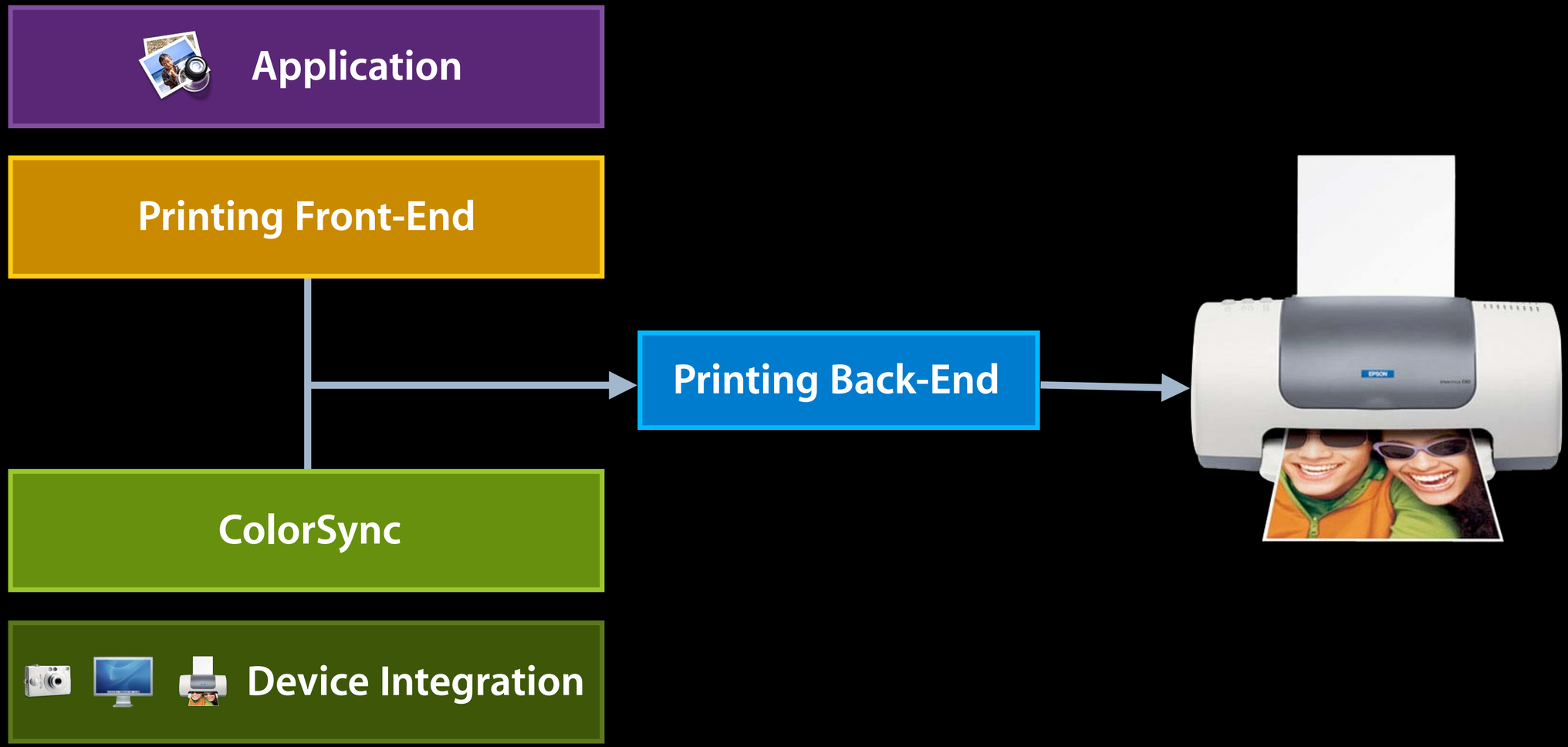
http://developer.apple.com/wwdc2005/

Printing Images

Basics

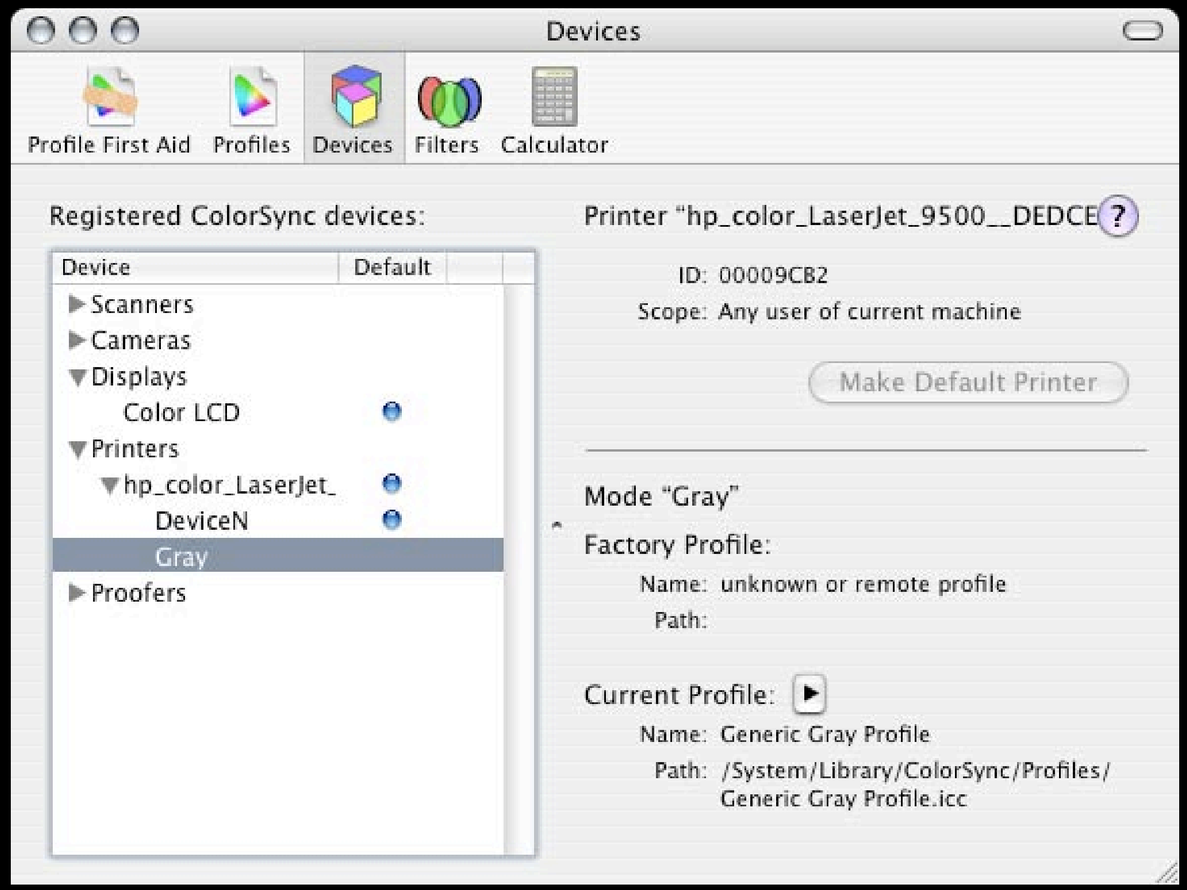
- Architecture overview
  - Printing as a part of Color Communication
- Application vs. Driver color management
- Controlling Printing Front-end
  - Preserving intended color information

Printing Architecture Overview



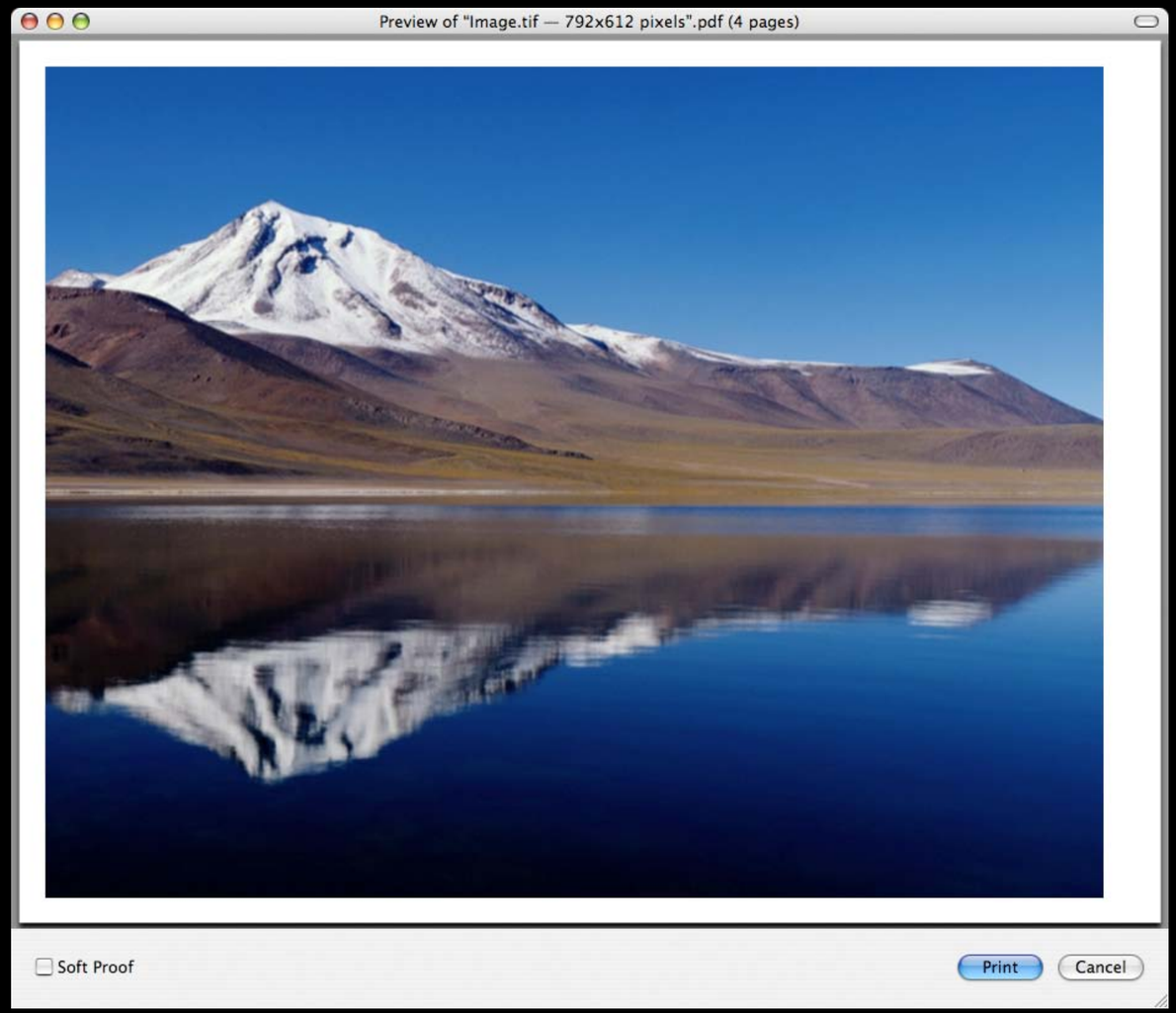
Printing Architecture Overview

Device integration database



Printing Images

Example of color communication based on ColorSync profiles

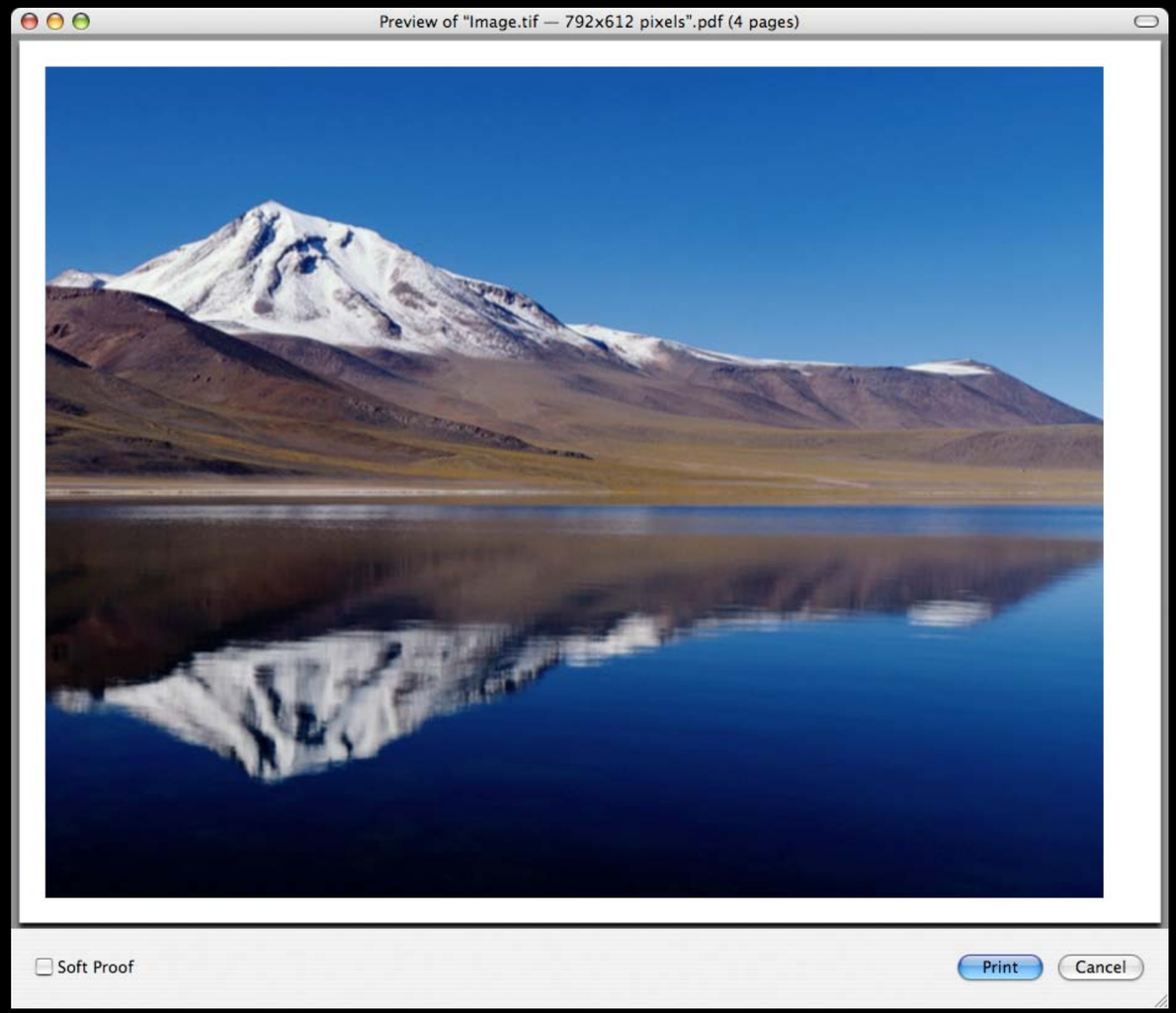


Printing Images

Example of color communication based on ColorSync profiles

Printing Images

Example of color communication based on ColorSync profiles



Printing Images

Color Communication

- Requires
  - Application to spool calibrated color data
  - Known printer profile for current print mode
  - Driver to not do any additional color manipulation
- Benefits
  - Allows for softproof in print preview
  - Allows savvy application to produce output specific color
    - PDF/X Output Intent

Printing Images

Application vs. driver color management

- (Application) ColorSync mode
  - Printing is a part of Color Communication
    - Color data is matched to the printer profile registered with ColorSync
    - Driver does not make any color adjustment
- Driver mode
  - Printing is excluded from Color Communication
    - Color data is matched to a generic profile and handed off to the Driver
    - Driver makes color adjustments which are not communicated to the system

Printing Images

Controlling printing front-end

```
- (NSPrintOperation*)
   printOperationWithSettings:(NSDictionary*) printSettings
                        error: (NSError **) outError
{
   [self calculateMargins: printSettings];

   PrintView* printView = [[self createPrintView] autorelease];

   NSPrintOperation* printOp =
       [NSPrintOperation printOperationWithView:printView
                                      printInfo:[self printInfo]];

   if (outError) // Clear error.
       *outError = NULL;

   return printOp;
}
```

Printing Images

Controlling printing front-end (Cont.)

```
- (void) drawRect: (NSRect) rect
{
   CGContextRef context = [[NSGraphicsContext currentContext] graphicsPort];

   CGAffineTransform imTransform = [self imageTransform:im];

   CGContextSaveGState(context);

   CGContextConcatCTM(context, imTransform);

   CGContextDrawImage(context, imageRect, image);

   CGContextRestoreGState(context);

   CGImageRelease(image);
}
```

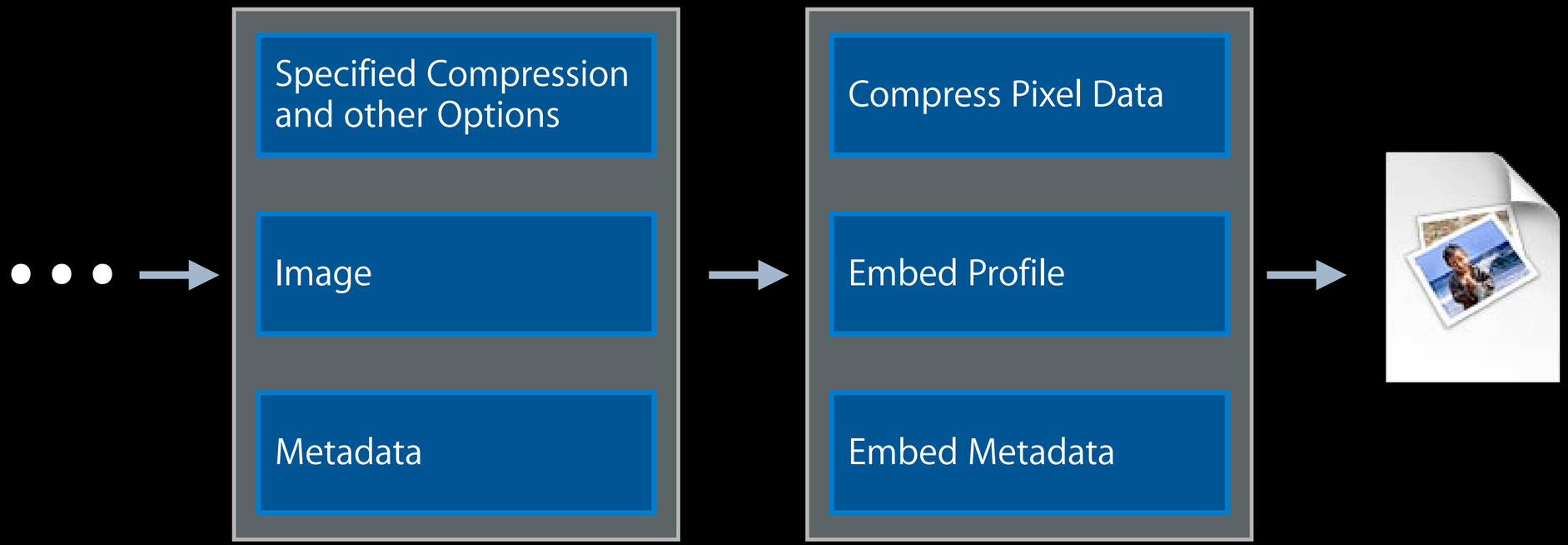
Saving Images With Image IO

David Hayward
Software Engineer

http://developer.apple.com/wwdc2005/

Saving Digital Images

General steps when saving an image



Saving Images With Image IO

CGImageDestination

- Create with
  - Use `CGImageDestinationCreateWithURL`
    or `CGImageDestinationCreateWithData`
    or `CGImageDestinationCreateWithDataConsumer`
  - Type identifier (file format)
  - Capacity (image count)
- Set properties for whole file: `CGImageDestinationSetProperties`
- Add each image with options and metadata: `CGImageDestinationAddImage`
- Finish by calling: `CGImageDestinationFinalize`

Saving Images With Image IO

CGImageDestination image and metadata

```
bool WriteJpegImage (CFURLRef url, CGImageRef image, float dpi)
{
   CGImageDestinationRef idst =
      CGImageDestinationCreateWithURL(url, CFSTR("public.jpeg"), 1, nil);

   CFStringRef keys[3]; CFNumberRef vals[3];

   keys[0] = kCGImageDestinationLossyCompressionQuality;
   vals[0] = CreateFloatNumber(0.80);

   keys[1] = kCGImagePropertyDPIWidth;   vals[1] = CreateFloatNumber(dpi);
   keys[2] = kCGImagePropertyDPIHeight;  vals[2] = CreateFloatNumber(dpi);

   CFDictionaryRef props = CFDictionaryCreate(nil, keys, values, 3,
      &kCFCopyStringDictionaryKeyCallBacks, &kCFTypeDictionaryValueCallBacks);

   CGImageDestinationAddImage(idst, image, props);
   return CGImageDestinationFinalize(idst);
}
```

Saving Images: Advanced Techniques

- Saving floating point images
- Adding save options to NSSavePanel

Saving Images: Advanced Techniques

Saving floating point images

- Just provide the `CGImageDestination` a `CGBitmap` with float pixels
- Currently only two file formats support writing float data
  - IEEE Float TIFF and OpenEXR

Saving Images: Advanced Techniques

Adding save options to NSSavePanel

- Start with options dictionary from original file
- Build file format pop-up from `CGImageDestinationCopyTypeIdentifiers`
- Use UI actions to store options and metadata into dictionary

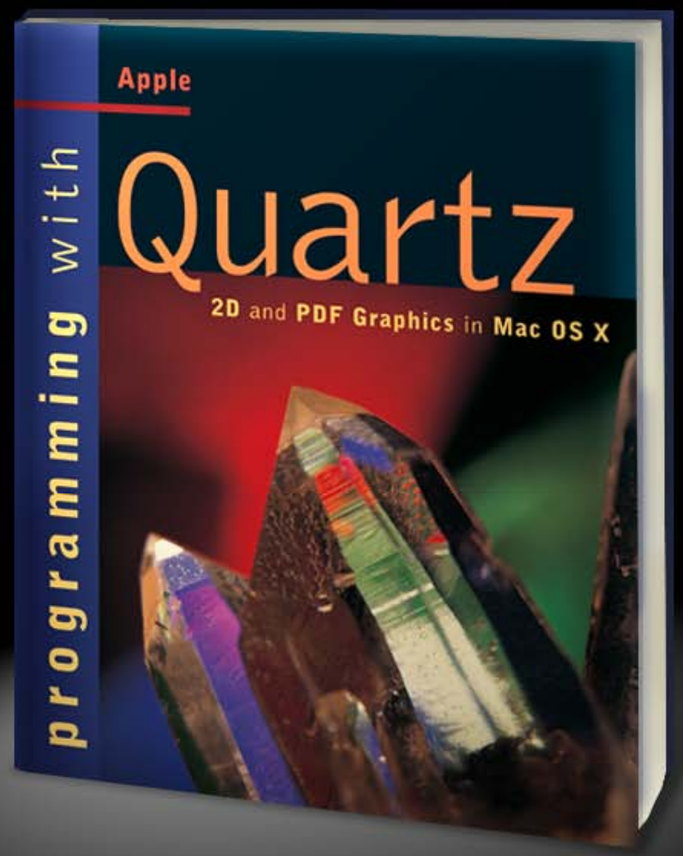
ImageApp Sample Code

http://developer.apple.com/wwdc2005/

Documentation, Sample Code, and Other Resources

http://developer.apple.com/wwdc2005/

Coming Soon From Morgan Kaufmann…



More info and sample chapter:
http://books.elsevier.com/mk/?isbn=0123694736

Related Sessions

Essential Ingredients for Mac OS X Imaging Solutions	Session 210 Wednesday 3:30PM
Printing Best Practices for Your Application	Session 212 Thursday 9:00AM
Core Image	Session 202 Tuesday 10:30AM
Optimal 2D Application Graphics	Session 216 Friday 10:30AM
Quartz 2D and PDF Kit	Lab Thursday 12–5:00PM

Q&A

Travis Brown
Graphics and Imaging Evangelist
travisb@apple.com

http://developer.apple.com/wwdc2005/

WWDC2005

Worldwide Developers Conference